An event-driven software architecture for process analysis in Additive Manufacturing

Matthias Moi*, Christian Lindemann*, Ulrich Jahnke*, Rainer Koch*

*Chair of Computer Application and Integration in Design and Planning (C.I.K.) and Direct Manufacturing Research Center (DMRC), The University of Paderborn, Pohlweg 47-49, 33098 Paderborn, Germany

Accepted August 16th 2013

Abstract

Additive Manufacturing is still not commonly accepted as a considerable manufacturing process for serial products. Build rate and cost estimation or even the traceability of parts concerning Quality Management issues is weak. That's also because of the unavailability of adequate software solutions. Self-made solutions with Spreadsheets are often hard to adapt and inflexible in usage. This paper presents a distributed event-driven software architecture for cost assessment and traceability from powder to finished products. Further the approach of event-driven processing for cost calculation following the activity based costing methodology is discussed. The methodology considers arbitrary events (e.g. machine data or market prices) that may have an effect for a more detailed process analysis.

Introduction

Additive Manufacturing (AM) is a fast growing technology in several domains like the aerospace industry. On the first of June 2013 the FP7 project RepAIR started successfully. The consortium performs research on future repair and maintenance for the aerospace industry using AM. The development of novel IT solutions is also a significant part of the project. The simultaneously increasing complexity of processes where decisions may influence quality and costs of parts, directly and indirectly as well, the need of information technology becomes important. The importance is also shown by the development of commercial solutions and products like Materialise Streamics™ (Dennis Vandenbussche 2012) or AutoFab. In order to specify costs more precisely, cost-drivers have to be identified. In (Lindemann et al. 2012) we presented a costing model, with focus on metal additive manufacturing (MAM). The methodology is focusing on the four main processes:

1. Build Preparation
2. Manufacturing
3. Post processing
4. Quality control

For these main processes we identified cost drivers like CAD-Preparation or machine preparation. Our approach mainly based on activity based costing, where more indirect costs are assigned into direct costs (Leitner 2007). In every main process IT can support engineers, designers and decision makers during the whole processes (1 to 4). From the IT point of view heterogeneous data from several sources (e.g. machine parameters or requirements on the part) has to be digitalized or synchronized in order to increase the traceability of a part during the whole processes. The traceability is also important for quality management processes. Through continuous gathering of data a continuous improvement of quality management and quality assurance processes can be achieved. Furthermore collected data may influence cost drivers or quality drivers in each process and therefore in each of their sub-process.

Based on these findings we are presenting an easy extendable software architecture that is based on the event-driven service oriented software architecture paradigm (cf. Mühl et al. 2006; Taylor 2009) and complex event processing (Etzion and Niblett 2011) to improve the traceability and simplify the cost estimation. Because of the fast development in additive manufacturing, a highly adaptable and configurable system with highly decoupled components is needed to keep in trying. The general implementation of the system as well as implementation details is not the subject of this work.

Event Processing

Even if the theory behind event processing is not new, the event-driven thinking enables us to compute complex issues by defining rules, usage of logical reasoning or pattern matching. An event may be everything that can happen in the real world. Etzion and Niblett defined this as follows:

"An event is an occurrence within a particular system or domain; it is something that has happened, or is contemplated as havening happened in that domain. The word event is also used to mean a programming entity that represents such an occurrence in a computing system" (Etzion and Niblett 2011).

Let's assume we would like to build a simple quality assurance alarm system for the observation of a build job. If there is an event type *LowTemperatureDetected* we can define a rule that sends an alarm to some system or tracks the information. Even this easy example shows how events can be processed. To get a little bit more complex, let's assume that we have the additional event type *BuildJobStarted*. By defining a simple rule we are able to define what should happen, if those two events occur. An example could be to stop the build job (which is also an event) and inform a responsible person in order to avoid wasting material. This example is illustrated in Figure 1. For each sequence of events arbitrary rules can be defined. Each performed action may generate further events. Therefore according to Etzion definition, everything that has happened can be interpreted as an event.

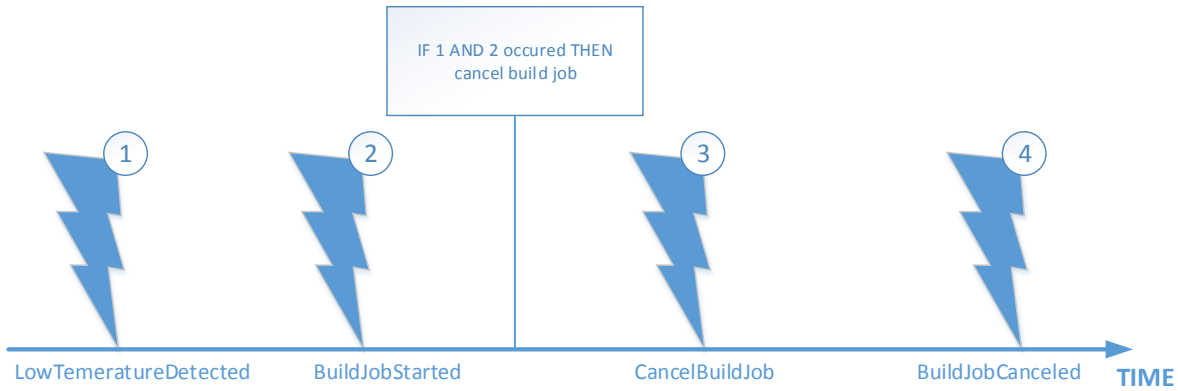

Figure 1 Event processing example

The process of events generating further events is called complex event processing. Each sequence of events from an event stream can also be interpreted as a scenario. In the example above the scenario can be described as follows: Detection of build jobs which started with a low temperature in the building chamber.

Event processing Applications

The general (abstract) structure of an event-driven application is shown in Figure 2. The Event producers (Sensors, AM-Machines, Human-Interaction) are the event generators. The

consumers are applications, systems or components consuming these events (e.g. displaying them). The producers may be hardware sensors (e.g. heat detectors) or even software components. Note, that an event producer may also be an event consumer. The Event Processing block asynchronously interconnects producers with consumers. In many event-driven applications this is done by a messaging system like a message bus where components send and receive event objects from the message bus. An event object is an instantiated object of a specific event type.

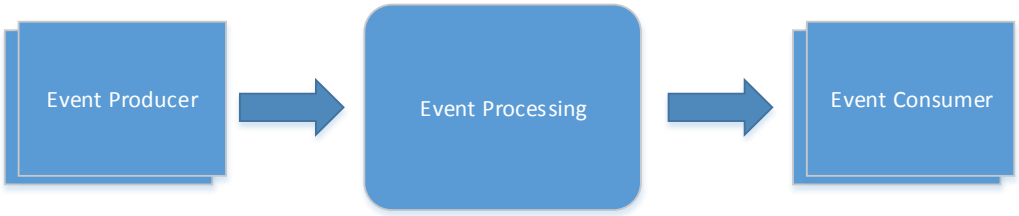

Figure 2 Event Processing

Summarizing (Luckham and Frasca 1998, S. 1) defined event processing as follows:

"[...] event processing is a new technology for extracting information from distributed message-based systems. This technology allows users of a system to specify the information that is of interest to them. It can be low level network processing data or high level enterprise management intelligence, depending upon the role and viewpoint of individual users. And it can be changed from moment to moment while the target system is in operation."

Event types

There are many existing types of events. In a software system they are called event types that have to be defined for a domain. Etzion and Niblett define event types as follows:

"[...] An event type is a specification for a set of event objects that have the same semantic intent and same structure; every event object is considered to be an instance of an event type." (Etzion and Niblett 2011)

Transported information by an event object is specified by a set of attributes in its event type definition. An attribute is a key-value pair. Attributes may be required or not. Event type description attributes like the event type identifier are required for each instanced event object. Generally attributes can be distinguished as follows: Header attributes, payload attributes and open content. Header attributes are the system defined attributes, e.g. for the identification of the event object. These attributes are mostly generated automatically by a system. Payload attributes containing the data to be transported, resp. the specific attributes to the event type. An example for the attribution of the *lowTemperatureDetected event type is shown in* Table 1 and Table 2.

Table 1 Header attributes of the event type *lowTemperatureDetected*

Attribute name (key)	**Attribute value (value)**
eventTypeIdentifier	*lowTemperatureDetected*
eventComposition	False
temporalGranularity	Second
occurrenceTime	08:02:04
eventIdentity	123456654321
eventSource	SLM250_1

Table 2 Payload attributes of the event type lowTemperatureDetected

Attribute name (key)	**Attribute value (value)**
temperatureUnit	*°F*
Temperature	*482*
desiredManufactoring Process	*1-0815*
occurrenceTime	*08:02:04*

Software components to support the above mentioned four main processes may run autonomously if required inputs and outputs are defined. After each step e.g. in the build preparation, new data gets available and can be used in arbitrary ways. The event-driven approach supports this thinking. Further the approach makes it easy to achieve a high decoupling between components. Dependency will only exist on required information (semantic level – inputs/outputs). E.g. components for the cost calculation in the post processing depend on the data of components supporting or observing the manufacturing process. Technically these components may work without their knowledge about the subsistence of each other. This makes the exchange of existing components as well as the adding simple.

Hence an event processing network can be build. It exists of event producers, event consumers and event processing agents (Etzion and Niblett 2011). The general idea of software agents relies on the definition of (Wooldridge 2002), were software agents are defined as independent computer systems interacting with their environment to achieve their goals. It also includes the communication with other agents. E.g. an event processing agent (EPA) may have the goal to store or process events, which are relevant to calculate the costs in the pre-processing, resp. the build preparation. EPA's acting autonomously and can be replaced by other agents or new versions of agents without the modification of other agents. In case of the fast developments in the AM domain, this is a very important fact, because you are very flexible in the extension and adaption of software components, resp. EPA.

Activity based costing with event-driven processing

In (Lindemann et al. 2012) we analyzed product lifecycle costs for a better understanding of cost drivers in AM. Our approach is based on the activity based costing methodology (ABC) (cf. Miller and Vollman 1985; Cooper R. and Kaplan R. S. 1988) . The general requirement to model event processing agents (EPA) is to have access to relevant data in order to calculate fixed costs, variable costs and overhead costs. Hence we assume that EPA have access to databases were material and costs related information is stored. The results of the applicability are shown in Table 3. Since the general assumption of Luckham, that everything that has happened can be an event, the application of the ABC methodology in event processing systems can be done.

Table 3 Applicability of event processing for ABC

Type of costs	Description	Applicability of event processing
Fixed costs / Overhead costs	Costs that are fixed in the whole process. E.g. costs for machine preparation or labor cost rates. Costs for administration etc.	The EPA must get access to relevant data. EPA must be able to update costs rates if necessary, e.g. due to environmental changes or changes at the market. By adding a new order/build job, events are generated, received by the agent and processed. All relevant data to estimate and calculate fixed and the proportional overhead costs are stored in databases (even distributed databases) and can be accessed from an EPA for further processing.
Variable costs	Costs for material, energy consumption during the manufacturing process,	EPA must be able to observe the processes. In the build part preparation. Decisions (e.g. material selection, build part orientation etc.) are

	machine cost rates.	taken by the designer/engineer. Each decision may affect the resulting costs. While modeling event types (e.g. machine or material selected), an EPA is able to process the data to pre-calculate the costs for the desired build job. Same holds for the manufacturing and the post processing. During and after the manufacturing events may influence the estimated costs.

Event-Driven Architecture in AM

In the previous sections we introduced how event-driven architectures can be designed with concept of event processing networks and event-driven agents. In order to support the whole process from powder to product we will now present the design of our event-driven architecture to achieve traceability and to calculate/estimate costs based on the ABC approach for the whole AM process.

The general architecture is divided into five main subsystems (see Figure 5). A subsystem encapsulates functionalities logically. Beside the event-driven approach the architecture follows the Model-View-Controller pattern, where the view (application subsystem) is updated by the model (storage subsystem, Logging and retrieval, message oriented middleware) through the controller (Agent Based Event Processing). The user interacts with the controllers indirectly.

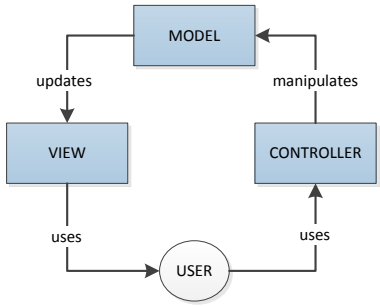

Figure 3 MVC pattern

Integration Subsystem

This subsystem coordinates the whole event-driven communication between the subsystems and integrates external systems like AM machines to get process data during the manufacturing. The Message Oriented Middleware (MOM) provides a message bus where subsystems, resp. software components, can subscribe arbitrary event types. Published event objects are forwarded to components that have a subscription for associated event types. The principle of MOM's event bus is illustrated in Figure 4.

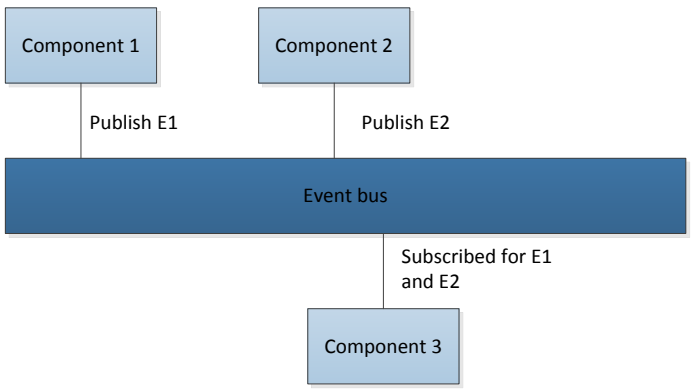

Figure 4 Event bus of the MOM

The communication via the MOM is asynchronous as by the definition of the messaging system via a message bus, resp. an event bus.

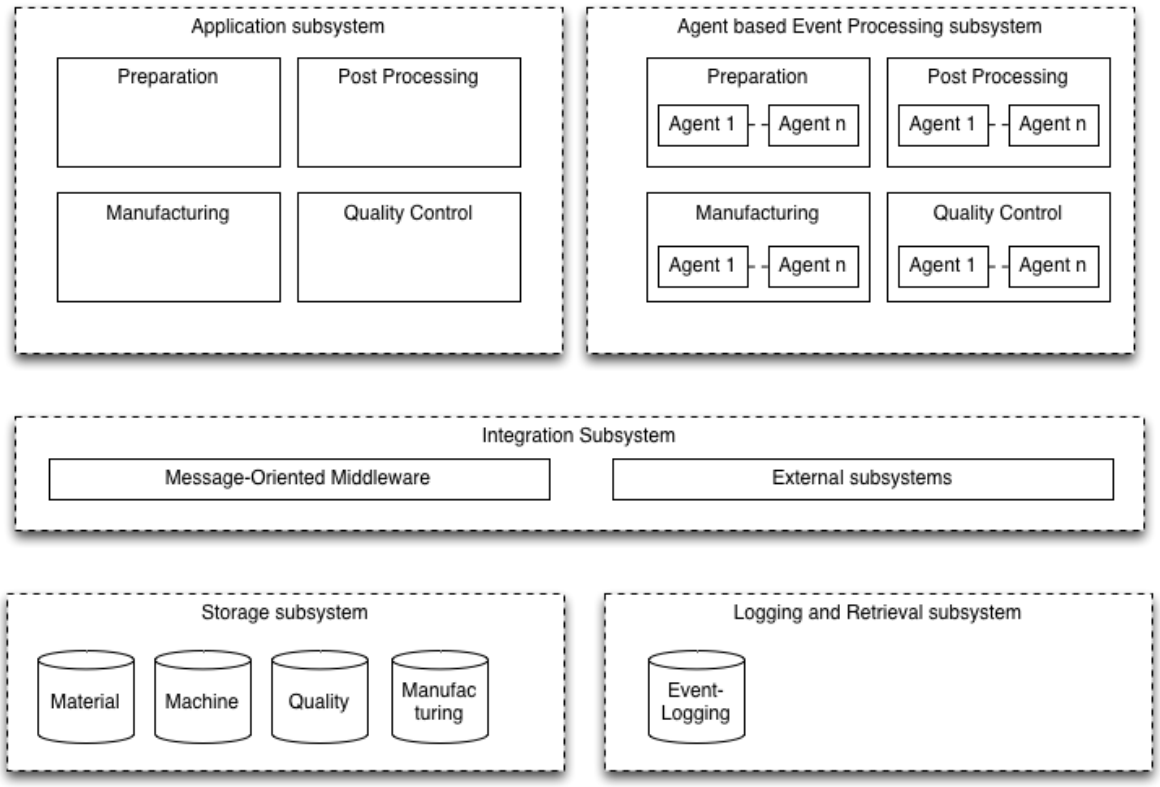

Figure 5 Event-driven architecture

Application Subsystem

The application subsystem is composed of four main components; the Preparation, the Manufacturing, the Post Processing and the Quality Control. The structure is based on the four identified main processes to estimate the product lifecycle costs during the whole AM process. The components in the application subsystem are designed to support the processes, especially the designers and engineers from powder to product. The main tasks of the components are listed below.

Preparation

- Definition of requirements relevant for the manufacturing process
- Selection of machine and material
- Definition of quality assumptions
    - Roughness
    - Surface quality…
- Visualization of energy consumption regarding the parts to be build
- Visualization of material consumption for support structures

Manufacturing

- Observe build jobs
- Show relevant events to the user
- Visualize actual vs. target performance by comparison of requirements and real-time data like machine data.

Post-Processing

- Observation of post processing activities like heat-treatment

- Visualization of estimated costs for post processing

Quality control

- Visualization of costs for quality control regarding specified requirements
- Documentation of new knowledge regarding quality control

The components in the application subsystem are representing the view, resp. the interface to the user. They are designed to consume events for information visualization for the user and to produce events by the interaction with the user. While interacting with the view, events are generated for further processing in the controller (Agent Based Event Processing subsystem).

Agent Based Event Processing

The Agent Based Event Processing subsystem covers the intelligence of the whole system. For each process specialized event processing agents (EPA) are defined. While components in the application subsystem are event producers and event consumers, EPA processing the events. This includes the generation of new, complex events by defined rules and pattern matching mechanisms. The main tasks of the EPA are listed below.

Preparation

- Cost Estimation in the preparation
- Calculation of estimated material and energy consumption regarding the specification done by the users and machine selection.
- Calculation of build time and time for post processing to rearrange parts in the building chamber.
- Estimation of energy consumption regarding the parts to be build
- Estimation of material consumption for support structures

Manufacturing

- Observe build jobs
- Processing of relevant events during the manufacturing process like critical situations recognized by defined rules or pattern matching.
- Visualize actual vs. target performance by comparison of requirements and real-time data like machine data.

Post-Processing

- Calculating of costs for post processing activities like heat-treatment
- Estimation of post processing costs
- Estimation of costs for support structure removal.

Quality control

- Estimation of costs for quality control and improvement of quality assurance methodologies regarding specified requirements.
- Analysis for process improvement by collected data.
- Correlation analysis between different build jobs for quality improvement and cost reduction.

Storage Subsystem and Logging/Retrieval

The storage subsystem covers databases (or even knowledge bases) to process and track all activates in the system. It builds the model of the whole system. The Storage subsystem provides an interface where components of the other subsystem can request and store data. The possible huge amount of data makes it possible to make further analysis like correlation analysis between chosen parameters in the build preparation.

The Logging and Retrieval subsystem is an event consumer which is subscribed for all event types at the MOM. This guaranties maximum event based traceability, because each event is stored. Event objects stored in the database can be reproduced for arbitrarily time frames.

.

Conclusion and future work

In this paper we've introduced the paradigm of event-driven architectures and showed the potential of application in the additive manufacturing domain. We've analyzed the applicability of event-driven processing for the activity based costing methodology. Based on these findings we designed an event-driven architecture to support the whole AM process in order to achieve a high traceability and an easier calculation of product-lifecycle costs by the usage of the event processing agents approach.

Regarding the architecture the next main steps will include be the implementation of the architecture based on existing software frameworks like ESPER[1], Oracle Complex Event Processing[2] or HornetQ[3].

Further investigations will be made in the identification and specification of process relevant event types and therefore the definition of event processing agents. Since everything that has been happed can be an event the challenge becomes which occurrences matter. Many occurrences throughout AM process are very process-dependent, which means that deep process analyses have to be made. The relevance of event types differs from process to process. E.g. usage of different materials may require different post processing steps were different events may occur or be relevant. Therefore we will focus on metal selective laser sintering processes in the initial phase.

This comes with research in rules and pattern for event-driven cost estimation of product-lifecycle costs in additive manufacturing. The aim is to define event-driven agents in order to derivate event-driven costing methodologies of existing activity based costing approaches.

[1] http://esper.codehaus.org/
[2] http://docs.oracle.com/cd/E13157_01/wlevs/docs30/pdf/get_started.pdf
[3] http://www.jboss.org/hornetq

Literature

Cooper R.; Kaplan R. S. (1988): Measure costs right: Make the right decisions.

Dennis Vandenbussche (2012): Complying with Quality and Traceability Requirements of the AM Process from Design to Production. How it is done in an Approved Medical Production, 16.06.2012.

Etzion, Opher; Niblett, Peter (2011): Event processing in action. Stamford CT u.a: Manning.

Leitner, A. (2007): Activity Based Costing: GRIN Verlag.

Lindemann, Christian; Jahnke, Urlich; Moi, Matthias; Koch, Rainer (2012): Analyzing Product Lifecycle Costs for a Better Understanding of Cost Drivers. In: *Solid Freeform Fabrication Symposium*.

Luckham, David C.; Frasca, Brian (1998): Complex event processing in distributed systems.

Miller, J. G.; Vollman, T. E. (1985): Hidden Factory: Harvard Business School Reprint.

Mühl, Gero; Fiege, Ludger; Pietzuch, Peter (2006): Distributed event-based systems. Berlin: Springer-Verlag.

Taylor, Hugh (2009): Event-driven architecture. How SOA enables the real-time enterprise. Upper Saddle River, NJ: Addison-Wesley.

Wooldridge, Michael (2002): Intelligent Agents: The Key Concepts: Springer Berlin / Heidelberg.